Analyzing Automata with Presburger Arithmetic and Uninterpreted Function Symbols

Vlad Rusu

▶ **To cite this version:**

Vlad Rusu. Analyzing Automata with Presburger Arithmetic and Uninterpreted Function Symbols. [Research Report] RR-4100, INRIA. 2001. inria-00072531

HAL Id: inria-00072531
https://inria.hal.science/inria-00072531

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Analyzing Automata with Presburger Arithmetic and Uninterpreted Function Symbols

Vlad Rusu, IRISA/INRIA Rennes

N˚4100

Janvier 2001

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4100--FR+ENG

Analyzing Automata with Presburger Arithmetic and Uninterpreted Function Symbols

Vlad Rusu, IRISA/INRIA Rennes

Thème 1 — Réseaux et systèmes
Projet PAMPA

Rapport de recherche n ° 4100 — Janvier 2001 — 16 pages

Abstract: We study a class of extended automata defined by guarded commands over Presburger arithmetic with uninterpreted functions. On the theoretical side, we show that the bounded reachability problem is decidable in this model. On the practical side, the class is useful for modeling programs with potentially infinite data structures, and the reachability procedure can be used for symbolic simulation, testing, and verification.

Key-words: program modeling, simulation, testing, verification.

(Résumé : tsvp)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Analyse d'automates étendus avec arithmetique de Presburger et fonctions non interprétées

Résumé : Nous étudions une classe d'automates étendus, définis par des commandes gardées exprimées en arithmetique de Presburger avec fonctions non interprétées. Nous montrons que le problme d'accessibilité bornée est décidable dans ce modèle. En pratique, ce formalisme permet de modéliser des programmes avec structures de données de taille potentiellement infinie, et la procédure d'accessibilité peut être utilisée pour la symulation symbolique, le test, et la vérification formelle.

Mots-clé : modélisation de programmes, simulation, test, vérification.

1 Introduction

Modern research in automated verification can be divided into two categories. In the first category, which includes languages and tools such as SPIN [12], SMV [3] and CADP [9], the focus is on using well-established techniques such as state enumeration or binary decision diagrams, and pushing them to the limits by optimizing the algorithms in every possible way. This has proven fruitful in terms of applications, and it is a mark of success of these techniques, which for the most part were initiated in the eighties and early nineties.

The second category of research is concerned with exploring new techniques, whose impact for practical applications, although in quantitative progress, still has to be assessed. Two representative examples for this category are Presburger arithmetic [19] and the theory of uninterpreted functions with equality [1]. Unlike finite-state methods, these relatively rich theories allow the modeling of real-life systems more accurately, without the need of abrupt simplifications. These theories are decidable (although their combination is not) and are efficiently implemented in state-of-the-art verification tools such as PVS [18] and Omega [15].

Motivated by these remarks, we investigate in this paper a class of extended automata that consist of transitions over a finite control structure, with guards and assignments in a decidable fragment of the theory of Presburger arithmetic with uninterpreted functions. The formalism is expressive (e.g., one transition can modify an unbounded number of function values). It can model quite naturally programs with unbounded data structures such as parametric-sized vectors and arrays.

We show that reachability in a bounded number of steps is decidable in this model. The result is interesting, in our opinion, because it gives a direct procedure for symbolic simulation. The procedure computes the weakest constraints on a program's input data for a given finite sequence of transitions to be executed. Thus, we obtain an automatic solution to the *test input problem* [16] from software testing: given a finite path in a program, find whether it is executable, and if this is the case, obtain input data for executing it. Then, given a coverage criterion [21] defined by a finite set of finite paths in the program (e.g., executing every instruction at least once, or checking each condition at least once), it is possible, using our procedure, to select the paths that are executable and to synthesize input data for executing them. This results in a complete coverage for the chosen coverage criterion. Finally, symbolic analysis can also be used as a semi-decision procedure for verification of safety properties: if a safety property does not hold, the procedure will detect this in a finite number of steps.

The rest of the paper is organized as follows. In Section 2 we present the theory of Presburger arithmetic with uninterpreted function symbols. In Section 3 we define a class of extended automata (called PF-automata) with guards and assignments in a decidable fragment of this theory. The initial condition may constrain an unbounded number of function values, and assignments may modify an unbounded number of such values. In Section 4 we present a decision procedure for the bounded reachability problem in PF-automata, based on symbolic analysis techniques. The procedure is implemented using the ICS decision procedure package [7] from SRI International. Section 5 presents conclusions, related work, and future work.

2 Background

In this section, we briefly describe the theory of Presburger arithmetic and its extension with uninterpreted function symbols.

2.1 Presburger Arithmetic

Let $\mathbb{Z}$ denote the set of integers, and V be a set of integer variables. A *term* is a finite, affine combination on the variables. An *inequality* is a comparison $(<, >, \leq, \geq, =)$ between terms. A *ground Presburger formula* is a finite Boolean combination of inequalities. A *Presburger formula* is a finite Boolean combination of inequalities, in which some variables can be quantified. Thus, if x, y, z, u are variables, then $x + 2y + 1$ is a term, $x \leq y \wedge x + 2y + 1 > 0$ is a ground Presburger formula, and $\forall z.x \leq y + z \wedge \exists u.x + 2y + u > y$ is a Presburger formula. Satisfiability in Presburger arithmetic is decidable [19], with time complexity triple exponential in the size of the formula [17] (but simple exponential in the ground fragment [2]).

2.2 Presburger Arithmetic with Uninterpreted Function Symbols

Let again V be a set of integer variables, and F be a set of function symbols. For each function $f \in F$, we only know its *arity* (a natural number n), and the fact that it is a function from $\mathbb{Z}^n$ to $\mathbb{Z}$. A *term with function symbols* is either a variable, or a function application to a term, or an affine combination of those. *Inequalities* (resp. *ground Presburger formulas*, resp. *Presburger formulas*) *with function symbols* are defined as in Section 2.1, except that they are built over terms with function symbols. Quantification over function symbols is not allowed. Thus, if x, y, z, u are variables and f, g are functions of arity one, $x + 2y + f(g(z))$ is a term with function symbols, $x \leq y \wedge x + 2y + f(g(z)) > 0$ is a ground Presburger formula with function symbols,

and $\forall x.x \leq y+f(g(z)) \wedge \exists u.x+2y+f(z) > g(u)$ is a Presburger formula with function symbols. In the sequel, we denote by PF the theory of Presburger arithmetic with function symbols. Satisfiability in PF is Σ_1^1-complete [13], but decidable in the ground fragment [22]. As we shall make use of the latter result, it is important to understand why it holds.

2.2.1 Decidability of the ground and existential fragments.

Shostak [22] made the following simple observation. In a ground formula with uninterpreted function symbols, the only relevant property about functions is that they map equals to equals, and, by instantiating this property to finitely many terms, it is possible to obtain an equivalent Presburger arithmetic formula. Let φ be a formula of the ground fragment of PF. For simplicity, we suppose that in φ there is only one unary function symbol f, which is only applied to two terms, t_1 and t_2. Then, φ is satisfiable if and only if the following Presburger formula is satisfiable:

$$\tilde{\varphi}: \quad \varphi[f(t_1)/f_1, f(t_2)/f_2] \wedge (t_1 = t_2 \supset f_1 = f_2) \tag{1}$$

That is, in $\tilde{\varphi}$, the function applications $f(t_i)$ from φ are replaced by new integer variables f_i, and the general property that function f maps equals to equals is instantiated to "equality of the terms t_i implies equality of the variables f_i". There is a one-to-one correspondence between the models of φ and those of $\tilde{\varphi}$.

Consider now the existential fragment of PF (i.e., only existential quantifiers are allowed, and under the scope of an even number of negations). Modulo a renaming of variables, it is possible to move all quantifiers to the outermost level. Then, a formula $\exists x.\varphi$ in this fragment has a model if and only if φ also has one. Indeed, if there exists values of the functions and free variables that satisfy φ, then the same values satisfy $\exists x.\varphi$, and if there exists a model for $\exists x.\varphi$, then this model, augmented with the "witness" value of x for the existential quantifier, is a model for φ. Thus, the existential fragment of PF is decidable. This reasoning can be pushed even further: a PF formula φ is satisfiable if and only if the formula $\exists f.\varphi$, where f is a function, is satisfiable. Note that the latter formula is not in PF. We will use this result in Section 4.

2.2.2 Dealing with universal formulas.

The universal fragment of PF consists of formulas in which only universal quantifiers are allowed, under an even number of negations. This fragment is highly undecid-

able[1]. Yet, universal formulas are useful: for example, specifying a property of *all* the elements of a parametric-sized vector requires a universal quantifier. In this section, we define a class of universal PF formulas that are satisfiable, and show how to decide satisfiability for combinations of ground and universal formulas. We use these results in the next sections.

We say formula ϕ is an *assignment to a function* f if ϕ is of the form $\forall i.(e_1 \supset f'(i) = e_2)$, where e_1 is a ground PF formula over $V \cup \{i\} \cup F$ (for some sets V of variables and F of functions such that $f' \notin F$), and e_2 is a term of PF over the same elements. Such formulas are satisfiable: for any values of the variables (including the quantified variable i) and functions, if e_1 holds, then $f'(i)$ can be chosen to be equal to e_2, otherwise, $f'(i)$ may have any value.

We say formula ψ of PF is *universally satisfiable* if the universal closure of ψ - the formula obtained from ψ by universally quantifying every variable of ψ - is satisfiable. For example, it is not hard to show that assignments to functions are universally satisfiable.

A formula of PF is *semi-universal* if it can be written as $\varphi \wedge \psi$, where φ is a ground formula, and ψ is a universally satisfiable formula of the universal fragment of PF with the property that each quantified variable is in the scope of a function symbol. (Note that this is not a syntactical definition, because the conjunct ψ is required to have a semantic property.)

The following lemma says that satisfiability of a semi-universal formula can be reduced to satisfiability of a ground formula, by instantiating universal quantifiers to finitely many terms.

Lemma 2.1 *Satisfiability in the class of semi-universal formulas is decidable.*

Proof. Let $\vartheta = \varphi \wedge \psi$ be a semi-universal formula. Without restricting the generality, we can assume that there is only one function symbol f occurring in ϑ. Formula ψ is then modified according to the following transformations: for every term in ψ of the form $f(t)$, where t may be any term except a quantified variable, ψ is transformed into the equivalent formula $\forall y.(y = t \supset \psi(t/y))$. Since we have started with a formula where all quantified variables are in the scope of a function symbol, we obtain after

[1] To see this, note that for any PF formula of the form $\forall x_1, \ldots x_n.\exists y.\varphi$, it is possible to skolemize y by replacing it with a new function symbol f, which gives the equivalent formula $\forall x_1, \ldots x_n.\varphi(y/f(x_1, \ldots x_n))$. Then, an arbitrary formula of PF can be translated into an equivalent universal formula by first moving all quantifiers to the outermost level, and then skolemizing the existentially quantified variables as shown above. In this way, the whole PF theory can be encoded in its universal fragment. Thus, satisfiability in the universal fragment of PF is Σ_1^1-complete.

all these transformations a formula ψ of the form $\forall z_1 \ldots \forall z_n . \hat{\psi}(f(z_1), \ldots, f(z_n))$, where $\hat{\psi}$ is a ground PF formula, and $f(z_1), \ldots, f(z_n)$ are all the occurrences of f in ψ. Let $f(t_1), \ldots, f(t_m)$ be all the occurrences of f in φ, where $t_1, \ldots t_m$ are terms. Then, ϑ is equivalent to:

$$\varphi \wedge \bigwedge_{z_1,\ldots z_n \in \{t_1,\ldots t_m\}} \hat{\psi}(f(z_1), \ldots, f(z_n)) \wedge \forall z_1, \ldots z_n \notin \{t_1, \ldots t_m\} . \hat{\psi}(f(z_1), \ldots, f(z_n))) \quad (2)$$

That is, formula ψ is split into two conjuncts: in the first one, the quantified variables $z_1, \ldots z_n$ are instantiated to every possible combination of the terms $t_1, \ldots, t_m$, and in the second one, the quantified variables $z_1, \ldots z_n$ are required to be different from all these terms. Finally, we show that the satisfiability of Formula (2) is equivalent to the satisfiability of its first two conjuncts, which constitute a ground PF formula, whose satisfiability is decidable:

$$\varphi(f(t_1), \ldots, f(t_m)) \wedge \bigwedge_{z_1,\ldots z_n \in \{t_1,\ldots t_m\}} \hat{\psi}(f(z_1), \ldots, f(z_n)) \quad (3)$$

Indeed, any model of Formula (2) is also a model for (3). Conversely, suppose (3) has a model, that is, values for the function f and the free variables appearing in (3). Any such model constrains the value of f only at the positions defined by the values of $t_1, \ldots, t_m$. Thus, for every position i different from these values, it is possible to choose the value of $f(i)$ freely. In particular, we choose these values as defined by the value of f in some model of the universal closure of ψ. (There exists such a model because we have supposed ψ is universally satisfiable). We consider a model of Formula (3) with f chosen as above, and show it is also a model of Formula (2). By construction, the chosen values of f and of the free variables will satisfy the first two conjuncts of Formula (2). These values also satisfy the third conjunct $\forall z_1, \ldots, z_n \notin \{t_1, \ldots t_m\} . \hat{\psi}(f(z_1), \ldots, f(z_n))$: since ψ is universally satisfiable, this is also the case for the weaker formula $\forall z_1, \ldots, z_n \notin \{t_1, \ldots t_m\} . \hat{\psi}(f(z_1), \ldots, f(z_n)))$. Thus, the chosen value for f, together with any values for the free variables (in particular, the ones chosen from the model of Formula (3)) will satisfy it. □

3 PF-automata

In this section we define the syntax and semantics of a class of extended automata with guards and assignments in a fragment of Presburger arithmetic with uninterpreted function symbols.

Definition 3.1 (PF-automaton) *A* PF*-automaton is a tuple* $\langle Q, q^0, V, F, \Theta, \mathcal{T} \rangle$*:*

- Q *is a finite set of locations,*
- $q^0 \in Q$ *is the initial location,*
- V *is a finite set of integer variables,*
- F *is a finite set of unary function symbols,*
- Θ *is a semi-universal* PF *formula, called the initial condition,*
- $\mathcal{T}$ *is a finite set of transitions. Each transition is a tuple* $\langle q, \gamma, \nu, \phi, q' \rangle$ *where*
  - $q \in Q$ *is the origin of the transition,*
  - γ *is a ground* PF *formula, called the guard of the transition,*
  - ν *is a finite set of variable assignments. An assignment to variable* $v \in V$ *is an expression of the form* $v' = r$*, where* r *is a term of* PF *over* V *and* F*. For each variable* $v \in V$*, there is in* ν *at most one assignment to* v*,*
  - ϕ *is a finite set of function assignments. An assignment to function* $f \in F$ *is an expression of the form* $\forall i.(e_1 \supset f'(i) = e_2)$*, where* e_1*,* e_2 *are respectively a ground* PF *formula and a term of* PF *over* $V \cup \{i\} \cup F$*. For each function* $f \in F$*, there is in* ϕ *at most one assignment to* f*,*
  - q' *is a location called the destination of the transition.*

Note that the initial condition Θ is required to be a semi-universal formula. As membership in this class is not decidable, other techniques (e.g., theorem proving) may be needed to establish that a given structure is a PF-automaton. We expect that PF-automata which model "real" programs will have rather simple initial conditions, whose satisfiability is not hard to assess.

Figure 1 is an example of PF-automaton, which models the successive insertion of all the elements of a vector g of size m into the sorted vector f of size n. The initial location is l_0, and the initial condition specifies that the vector f is sorted. Variable j is used as a counter going over the elements of vector g. On the transition from l_0 to l_1, the function assignment to f specifies that all the elements of f that are greater than $g(j)$ are shuffled one position to the right. The function assignment on the transition from l_1 to l_2 specifies the insertion of the new element $g(j)$ at the correct position. Because f is sorted, this position is unique. Then, the size of f grows by one, and the next element of g is processed.

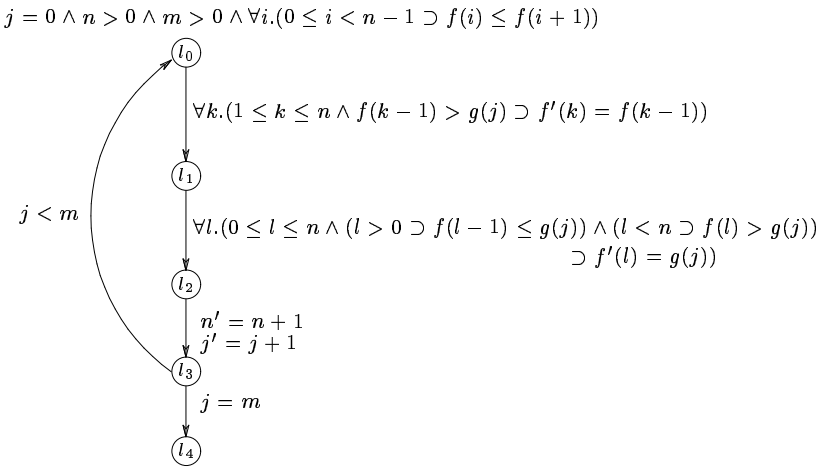

Figure 1: Example of PF-automaton: Insertion in Sorted Vector

PF-automata are useful for modeling programs with potentially unbounded data structures such as files, buffers, and arrays of parametric size. In Definition 3.1, we have assumed that the only basic type is integer, but most other types (Booleans, enumerations, records, subranges) can be encoded using integers. The restriction that there is at most one assignment for each variable and function application is useful for avoiding semantic complications (i.e., situations where a function gets two different values simultaneously). It can be dealt with in practice by introducing new transitions to sequentialize the assignments. Also, the restriction that all functions are unary is used to simplify semantic definitions.

3.0.3 Semantics of PF-automata.

A *valuation* is a mapping that assigns, to each free variable appearing in the automaton, a value in $\mathbb{Z}$, and to each function symbol, a function from $\mathbb{Z}$ to $\mathbb{Z}$. We denote by $\mathcal{V}$ the set of all valuations. A *state* is a pair (q, v) consisting of a location $q \in Q$ and a valuation $v \in \mathcal{V}$. Note that, for a PF-automaton with at least one function symbol, there are uncountably many states. An *initial state* is a state of the form (q^0, v^0) such that $v^0 \models \Theta$, that is, the location is initial and the values of the variables and functions satisfy the initial condition Θ. The set of states is denoted by $\mathcal{S}$, and the set of initial states is denoted by $\mathcal{S}^0$. Each transition $\tau \in \mathcal{T}$ defines a *transition relation* $\varrho_\tau \subseteq \mathcal{S} \times \mathcal{S}$, in the following way. Intuitively, s and s' are in the

relation ϱ_τ if the location of s (resp. of s') is the origin (resp. destination) of τ, and the variables and functions in s satisfy the guard of τ. Moreover, the variables and functions get new values according to the assignments of τ

Formally, for a PF formula φ and a valuation $v \in \mathcal{V}$, let $\varphi[v]$ be the truth value of φ when the free variables and the function symbols of φ evaluate according to v. For a term r, we denote by $r[v]$ the integer value obtained by evaluating variables and function symbols according to v. We now define how valuations are modified by assignments. According to Definition 3.1, these are of two kinds: assignments to variables and assignments to functions. Let $v' \in \mathcal{V}$ be the valuation obtained from v after an assignment, then, v' is obtained in the following way. If the assignment is of the form $x' = r$, where x is a variable, then v' is the valuation such that for all $u \in F \cup (V \setminus \{x\})$, $v'(u) = v(u)$, and $v'(x) = r[v]$. Otherwise, the assignment is of the form $\forall i.(e_1 \supset f'(i) = e_2)$, where e_1 is a ground formula and e_2 is a term, both over the variables $V \cup \{i\}$ and functions F. Then, v' is the valuation such that for all $u \in V \cup (F \setminus \{f\})$, $v'(u) = v(u)$, and $v'(f)$ is defined as follows: for any $i_0 \in \mathbb{Z}$, let $e_1[v, i/i_0]$ (resp. $e_2[v, i/i_0]$) denote the value of e_1 (resp. e_2) when the free variables V evaluate according to v, and i equals i_0. Then, for any $i_0 \in \mathbb{Z}$ such that $e_1[v, i/i_0]$ holds, $v'(f)(i_0) = e_2[v, i/i_0]$, and $v'(f)(i_0) = v(f)(i_0)$ otherwise. Finally, for $\tau = \langle q, \gamma, \nu, \phi, q' \rangle$ a transition, we denote by $v[\nu, \phi]$ the valuation obtained by successively transforming v according to the assignments of τ. (Note that the order in which this is done is not important, because there is at most one assignment per variable or function symbol). Then, the transition relation ϱ_τ of transition $\tau = \langle q, \gamma, \nu, \phi, q' \rangle$ is the smallest relation defined by the following rule:

$$\frac{s, s' \in \mathcal{S},\ s = (q, v),\ s' = (q', v'),\ \gamma[v] = \mathit{true},\ v' = v[\nu, \phi]}{\varrho_\tau(s, s')}$$

We denote by $\varrho = \bigcup_{\tau \in \mathcal{T}} \varrho_\tau$ the transition relation of the PF-automaton. A *run* is a sequence of states: $\rho : \quad s_0, s_1 \ldots, s_n$ such that $s_0 \in \mathcal{S}^0$, and for $i = 0 \ldots n-1$, $\varrho(s_i, s_{i+1})$ holds. The *length* of run $\rho : \quad s_0, s_1 \ldots, s_n$ is n. A state $s \in \mathcal{S}$ is *reachable in at most m steps* if there exists a run of length n ($n \leq m$) whose last state is s. The *bounded reachability problem* is: given a set of states Ω and an integer m, is there a state $s \in \Omega$ which is reachable in at most m steps. In Section 4 we prove that this problem is decidable for PF-automata.

4 Symbolic Analysis

For a transition τ and an arbitrary predicate ϑ on states, the predicate $post_\tau(\vartheta)$ characterizes the states s' that can be reached by taking transition τ from some state s satisfying φ:

$$post_\tau(\vartheta): \quad \exists s.\varrho_\tau(s, s') \wedge \vartheta(s).$$

For a sequence of transitions $\sigma : \quad \tau_1, \ldots, \tau_n$ and a predicate ϑ, the predicate $post_\sigma(\vartheta)$ is defined as $post_\sigma(\vartheta)$: $post_{\tau_n}(post_{\tau_{n-1}}(\ldots post_{\tau_1}(\vartheta)))$. In the sequel, we identify sets of states with the formulas that characterize them, and let the set of states Ω be a ground PF formula. Clearly, Ω is reachable in at most m steps if and only if there exists a sequence σ of contiguous transitions, of length at most m, and starting in the initial location, such that $post_\sigma(\mathcal{S}^0) \cap \Omega \neq \emptyset$.

Since there are finitely many sequences of transitions up to a given length, it is enough to show that, for any finite sequence σ of contiguous transitions starting in the initial location, the formula $post_\sigma(\mathcal{S}^0) \wedge \Omega$ is in a class where satisfiability is decidable. We show by induction on the length of σ that $post_\sigma(\mathcal{S}^0)$ is of the form $\exists x_1, \ldots x_k \exists f_1, \ldots f_l.\vartheta$, where $x_1, \ldots x_k$ are variables, $f_1, \ldots f_l$ are functions, and ϑ is a semi-universal formula (cf. Section 2.2). Thus, satisfiability of $post_\sigma(\mathcal{S}^0)$ is reduced to satisfiability of ϑ, which is decidable (cf. Lemma 2.1).

The base step is obvious: by Definition 3.1 of PF-automata, the initial condition Θ is a semi-universal formula, which is a particular case of the desired form.

For the inductive step, it is enough to show that, for any transition $\tau = \langle q, \gamma, \nu, \phi, q' \rangle$ of the PF-automaton and any formula Ψ of the form $\Psi : \quad \exists x_1, \ldots x_k \exists f_1, \ldots f_l.\vartheta$ with ϑ a semi-universal formula, the formula $post_\tau(\Psi)$ is of the same form. Without restricting the generality, we assume that the PF-automaton has only one function symbol f, and that the function assignments ϕ of transition τ consists of one element, of the form $\forall i.(e_1 \supset f'(i) = e_2)$. Let $\overline{x}$ denote the variables of the PF-automaton, and $\nu(\overline{x})$ denote the effect of the variable assignments ν on the variables. Then, $post_\tau(\Psi)$ can be written as the following formula over the next-state variables $\overline{x}'$ and next-state function f':

$$post_\tau(\Psi): \ \exists f.\exists \overline{x}.(\overline{x}' = \nu(x) \wedge \Psi \wedge \forall i.(e_1 \supset f'(i) = e_2) \wedge \forall i.(\neg e_1 \supset f'(i) = f(i))) \tag{4}$$

By moving all the existential quantifiers in Ψ at the outermost level of the Formula (4), we obtain that $post_\tau(\Psi)$ is equivalent to the following formula:

$$\exists \overline{x}, x_1, \ldots, x_k.\exists f, f_1, \ldots f_l.(\overline{x}' = \nu(x) \wedge \vartheta \wedge \forall i.(e_1 \supset f'(i) = e_2) \wedge \forall i.(\neg e_1 \supset f'(i) = f(i))) \tag{5}$$

What we still have to show is that the formula obtained from (5) after removing all existential quantifiers is a semi-universal formula. By induction hypothesis, ϑ is a semi-universal formula. Thus, $\vartheta = \varphi \wedge \psi$, where φ is a ground PF formula, and ψ is a universally satisfiable formula of the universal fragment of PF (cf. Section 2.2). Hence, it is possible to write the formula obtained from (5) after removing all existential quantifiers, as the conjunction $\varphi' \wedge \psi'$, where $\varphi' : \quad \overline{x}' = \nu(x) \wedge \varphi$, and $\psi' : \quad \psi \wedge \forall i.(e_1 \supset f'(i) = e_2) \wedge \forall i.(\neg e_1 \supset f'(i) = f(i))$.

Clearly, φ' is a ground PF formula, and ψ' is a universal PF formula with the property that all quantified variables are in the scope of a function symbol (cf. Section 2.2). To complete the proof, we just have to show that ψ' is universally satisfiable. By induction hypothesis, ψ is universally satisfiable, thus, in particular, there exists a value of f such that, for any values of the variables $\overline{x}$, ψ is satisfied. Let the value of f' be defined as follows: for all values of i and of the variables $\overline{x}$, if e_1 holds, then the value of $f'(i)$ equals that of e_2, otherwise the value of $f'(i)$ equals that of $f(i)$. By adding the value of f' to the model of the universal closure of ψ, we obtain a model for the universal closure of ψ'. Hence, ψ' is universally satisfiable. □

Discussion.

The above proof shows that checking reachability in m steps involves checking satisfiability of a semi-universal formula ϑ with m copies of each variable and function symbol. This, in turn, involves instantiating every universal quantifier from the universal part of ϑ to all the terms in its ground part (cf. Lemma 2.1). Finally, a decision procedure for ground Presburger arithmetic with uninterpreted function symbols is used to decide the resulting ground formula. We use the ICS decision procedure package from SRI International [7].

Preliminary results with our symbolic analysis prototype are encouraging: for example, a symbolic simulation of a path of about ten thousand steps in a vector-sorting algorithm was completed in about twenty hours. This means ten thousand calls to the decision procedures for checking formulas with thousands of variables and function applications. As optimizations in both ICS and our prototype are still being developed, we expect to be able in the future to perform symbolic simulation on real-size programs and specifications.

Finally, it is worth noting that a simple extension of PF-automata which consists in letting the guards be universal PF formulas, is too expressive for symbolic simulation, as reachability even in one step becomes highly undecidable (cf. Section 2.2).

5 Conclusion, Related Work, and Future Work

In search of new infinite-state models for which some verification problems are still solvable, we investigate in this paper a class of extended automata, called PF-automata, with guards and assignments in a decidable fragment of Presburger arithmetic with uninterpreted function symbols. This formalism allows to model quite naturally programs with unbounded data structures such as parametric-sized vectors and arrays. The model is expressive: the initial condition may constrain an unbounded number of function values, and assignments may modify an unbounded number of such values. The latter can be seen as meta-transitions, which encode in one step the execution of an unbounded number of transitions. We present a decision procedure for the bounded reachability problem in this model. The procedure works by symbolically simulating the initial states over finite sequences of transitions of the automaton. It is implemented in `caml` and uses the ICS decision procedures from SRI International [7]. Symbolic simulation has a number of practical applications:

- it is a useful technique for understanding and debugging programs by interactive execution,
- it can be employed for performing accurate structural testing: given a coverage criterion, which is a finite set of finite paths in the program (as computed by, e.g., a commercial tool [6]), our procedure computes input data for executing those paths that are executable and discard those that are not, resulting in a complete coverage for the chosen criterion,
- finally, symbolic analysis can also be used as a semi-decision procedure for formal verification of safety properties: if a safety property does not hold, our reachability procedure detects this, otherwise, it will loop forever.

Related Work.

The literature on the analysis of automata extended with integer variables is vast; see [4] for new results and as an entry point. Concerning automata with uninterpreted function symbols, a recent result [8] shows that simulation is decidable for certain classes of such automata. The only results we are aware of about automata extended with Presburger arithmetic *and* function symbols are reported in [15]. Here, the Omega tool is used to analyze graphs whose edges are annotated by ground formulas in this logic. A *reachable* operation computes an over-approximation of the reachable

states. The operation will work for graphs with either cycles *or* functions, but not both.

The other main body of related work is structural testing. Most commercial tools (e.g., [6]) can measure the coverage with respect to a given coverage criterion for input data provided by the user. Moreover, the paths in the criterion are not always executable. Research on synthesizing input data using symbolic simulation was started in the seventies [20, 5] and has received renewed attention recently [11]. However, these techniques are currently limited to programs with scalar data types (vectors and arrays are not treated). Some techniques for dealing with vectors and arrays have been proposed [14, 10]. To our knowledge, our model, which allows an unbounded number of values to be constrained by the initial condition and the assignments, is among the most expressive for which symbolic simulation techniques exist.

Future Work.

The main direction of future work consists in optimizing the prototype symbolic analysis tool and adding new features to it, such as structural path computation for several coverage criteria, dependency analysis of variables, and slicing. We are also interested in using symbolic analysis techniques for obtaining coverage measures in conformance testing.

Acknowledgments.

Thanks to Duncan Clarke, Philippe Darondeau, Thierry Jéron, and John Rushby for useful comments and suggestions, and to Harald Ruess for help with ICS.

References

[1] W. Ackerman. *Solvable Cases of the Decision Problem.* North-Holland Publishing Company, Amsterdam, 1954.

[2] W. Bledsoe. A new method for proving certain Presburger formulas. In *4th International Joint Conference on Artificial Intelligence*, Tbilissi (USSR), pp 15–21.

[3] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill and J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[4] T. Bultan, R. Gerber, and W. Pugh. Model checking concurrent systems with unbounded integer variables: symbolic representations, approximations, and experimental results. *ACM Transactions on Programming Languages and Systems*, 21(4): 747–789, 1999.

[5] L.A. Clarke. A system to generate test data and symbolically execute programs. *IEEE Transactions on Software Engineering*, 2(3): 215–22, 1976.

[6] S. Cornett. Code coverage analysis. Available at `www.bullseye.com/coverage.html`

[7] D. Cyrluk, P. Lincoln, and N. Shankar. On Shostak's decision procedure for combinations of theories. *Computer-Aided Deduction, CADE '96*, LNAI 1104, pp 463–477, 1996.

[8] W. Damm, A. Pnueli and S. Ruah. Herbrand automata for hardware verification. *Conference on Concurrency Theory (CONCUR'98)*, LNCS 1466, pp 67–83, 1998.

[9] J-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu CADP: A protocol validation and verification toolbox. *Computer-Aided Verification, CAV'96*, LNCS 1102, pp 437–440, 1996.

[10] A. Goldberg, T.-C. Wang, and D. Zimmerman Applications of feasible path analysis to program testing. *International Symposium on Software Testing and Analysis, ISSTA'94*, pp 80–94, 1994.

[11] E. Gunter and D. Peled. Path exploration tool. *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'99*, LNCS 1579, pp 405–479.

[12] G.J. Holzmann. *Design and Validation of Communication Protocols.* Prentice Hall, 1991.

[13] J. Halpern. Presburger arithmetic with uninterpreted function symbols is Π^1_1-complete. *Journal of Symbolic Logic*, 56:637–642, 1991.

[14] R. Jasper, M. Brennan, K. Williamson, and D. Zimmerman. Test data generation using feasible path analysis. *International Symposium on Software Testing and Analysis ISSTA'94*, pp 95–107, 1994.

[15] W. Kelly, V. Maslov, W. Pugh, E. Rosser, T. Shpiesman, and D. Wonnacott. The Omega library interface guide. Available at `www.cs.umd.edu/projects/omega`.

[16] G. J. Myers. The Art of Software Testing. John Wiley and Sons, 1979.

[17] D. Oppen. A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger arithmetic. *Journal of Computer and System Sciences*, 16(3):323–332, 1978.

[18] S. Owre, J. Rushby, N. Shankar, and F. von Henke. Formal verification for fault-tolerant architectures: prolegomena to the design of PVS. *IEEE Transactions on Software Engineering*, 21(2):107–125, 1995.

[19] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zählen in Welchem die Addition als einzige Operation hervortritt. Sprawozdanie z I Kongresu Matematykow Krajow Slowcanskich Warszawa, Poland, 1929, pp 92–101.

[20] C. Ramomoorthy, S. Ho, and W. Chen. On the automated generation of program test data. *IEEE Transactions on Software Engineering*, 2(4):293–300, 1976.

[21] S. Rapps and E. Weyuker. Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, 11(4):367–375, 1985.

[22] R. Shostak. A practical decision procedure for arithmetic with uninterpreted function symbols. Journal of the ACM, 26(2):351–360, 1979.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399